IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Attorney Docket No.: **RAL920000123US1**

In re Application of:	§	
	§	
GARVEY	§	
	§	Examiner: **STEELMAN, M.**
Serial No.: **09/876,366**	§	
	§	Art Unit: **2122**
Filed: **7 JUNE 2001**	§	
	§	
For: **IF STATEMENT HAVING AN EXPRESSION SETUP CLAUSE TO BE UTILIZED IN STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING**	§	

APPEAL BRIEF

MS Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The present Brief is submitted in support of the Appeal in the above-identified application.

Please charge IBM Corporation Deposit Account **50-0563** in the amount of $500.00 for the submission of the present Brief. No additional fee or extension of time is believed to be required; however, in the event an additional fee or extension of time is required, please charge that fee to the IBM Corporation Deposit Account **50-0563**.

CERTIFICATE OF FACSIMILE TRANSMISSION
37 CFR § 1.8(a)

I hereby certify that this correspondence is being transmitted to the United States Patent and Trademark Office via facsimile on the date below.

2/15/05
Date

Vicky Silvaggi
Signature

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The present application is assigned to International Business Machines Corporation, the real party of interest.

RELATED APPEALS AND INTERFERENCES

No related appeal is presently pending.

STATUS OF THE CLAIMS

Claims 8-23, which were finally rejected by the Examiner as noted in the Final Office Action dated November 30, 2004 and in the Advisory Action dated February 8, 2005, are being appealed.

STATUS OF AMENDMENTS

A Response was submitted on December 10, 2004 in reply to the Final Office Action dated November 30, 2004.

SUMMARY OF THE CLAIMED SUBJECT MATTER

As recited in Claim 8 (and similarly in Claim 15), a state machine for handling a structured assembly language IF construct includes an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state (as shown in Figure 2). In response to a recognition of a SETUP_IF clause, the state machine transitions from the IF state or the ELSE_IF state to the SETUP_IF state. In response to a recognition of an ELSE_IF clause, the state machine transitions from the SETUP_IF state to the ELSE_IF state.

In Claim 22 (and similarly in Claim 23), an assembler (such as an assembler **30** in Figure **3**) residing in a data processing system for processing structured assembly language implements a state machine having an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state. The assembler includes means for identifying a SETUP_IF clause (such as a lexer **32** in Figure **3**). After associating the identified SETUP_IF clause with an ELSE_IF clause having a test condition, the assembler inserts instructions from the identified SETUP_IF clause

prior to the test condition of the ELSE_IF clause where the ELSE_IF clause logically follows a prior IF clause or a prior ELSE_IF clause (*see* parser 33 in Figure 3).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The Examiner's rejection of Claims 8-23 under 35 U.S.C. § 103(a) as being unpatentable over *Mead* (US 4,099,230).

ARGUMENT

The Examiner's rejections of Claims 8-23 are not well-founded and should be reversed.

I. *Mead* is not related to structured assembly programming

According to MPEP § 2111.02, the preamble needs to be given effect of a limitation when it "breaths life and meaning into the claim" and is "essential to point out the invention defined by the claim." The preambles of Claims 8 and 23 recite "[a] computer program product .... for processing structured assembly language" (emphasis added). Such recitation breaths life and meaning into the claims and puts the recitations of "in response to recognizing a SETUP_IF clause" and "in response to recognizing a ELSE_IF clause" in the body of the claims in context. Without the preamble, a reader may not be able to realize that the SETUP_IF and ELSE_IF clauses are specifically related to a structured assembly programming.

Furthermore, the preamble of Claim 15 recites "an assembler for processing structured assembly language." The preamble of Claim 22 recites "[a]n assembler ... for processing structured assembly language." Thus, the preambles of both Claims 15 and 22 are essential to point out the invention defined by the claim as an assembler and not simply a generic application software.

As such, the preambles of Claims 8, 15 and 22-23 do not merely recite the purpose of a process because the body of those claims depends on its respective preamble for completeness. Because *Mead* does not teach or suggest "structured assembly language" at all; thus, *Mead* is not applicable for the purpose of the § 103 rejection.

II. *Mead* does not teach or suggest the claimed SETUP_IF state and its transitions

Claim 8 (and similarly Claims 15 and 22-23) recites "a state machine having an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state" (lines 3-4). Thus, the claimed state machine has five separate and distinctive states, namely, an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state.

On page 5 of the Final Office Action, the Examiner states that the claimed SETUP_IF state is not disclosed by *Mead*, but then the Examiner asserts that the claimed SETUP_IF state is merely another reserved term related to a label created by a programmer. It is well-known in the art that each state within a state machine is more than merely a label. Thus, contrary to the characterization by the Examiner, the claimed SET_UP state is not merely a label created by a programmer.

To support her point, the Examiner has attached the definition of a state machine from Google.com along with the Advisory Action. According to Google.com, a state machine "specifies the sequences of states that an object or an interaction goes through during its life in response to events, together with its responses actions." A state machine allows a program to chill "at a certain state for a while, doing whatever is specified at that state, until it is told to move on to a different state by instructions in the current state or an external stimulus." Appellant agrees.

Claim 8 (and similarly Claim 15) recites "program code means for transitioning from said IF state or said ELSE_IF state to said SETUP_IF state, in response to recognizing a SETUP_IF clause" (lines 5-6) and "program code means for transitioning from said SETUP_IF state to said ELSE_IF state, in response to recognizing an ELSE_IF clause" (lines 7-8). Thus, according to the claimed invention, in response to a recognition of the SETUP_IF clause, the state changes from the IF state or the ELSE_IF state to the SETUP_IF state. Then, in response to a recognition of the ELSE_IF clause, the state changes from the SETUP_IF state to the ELSE_IF state.

Although Mead discloses various programming instructions or clauses, *Mead* does not teach or suggest the claimed step of "transitioning from said IF state or said ELSE_IF state to said SETUP_IF state, in response to recognizing a SETUP_IF clause" and the claimed step of "transitioning from said SETUP_IF state to said ELSE_IF state, in response to recognizing an ELSE_IF clause" in Claims 8 and 15. Because the claimed invention recites novel features that are not taught or suggested in *Mead*, the § 103 rejection is improper.

III. *Mead* does not teach or suggest an assembler for processing structured assembly language

Claim 22 (and similarly Claim 23) recites an assembler for processing programming instructions written in structured assembly language, and the assembler includes "means for inserting instructions from said identified SETUP_IF clause prior to the test condition of said ELSE_IF clause where said ELSE_IF clause logically follows a prior IF clause or a prior ELSE_IF clause" (lines 8-10).

As mentioned above, *Mead* is not related to structured assembly language programming, and hence does not teach or suggest an assembler for processing structured assembly language. Furthermore, since *Mead* does not disclose a SETUP_IF clause (or its equivalent), *Mead* does not teach or suggest any means that is capable of "inserting instructions from said identified SETUP_IF clause prior to the test condition of said ELSE_IF clause where said ELSE_IF clause logically follows a prior IF clause or a prior ELSE_IF clause," as recited. Thus, the § 103 rejection is improper.

CONCLUSION

For the reasons stated above, Appellant believes that the claimed invention clearly is patentably distinct over the cited references and that the rejections under 35 U.S.C. § 103 are not well-founded. Hence, Appellant respectfully urges the Board to reverse the Examiner's rejection.

Respectfully submitted,

Antony P. Ng
Registration No. 43,427
DILLON & YUDELL, LLP
8911 N. Cap. of Texas Hwy., suite 2110
Austin, Texas 78759
(512) 343-6116

ATTORNEY FOR APPELLANT

CLAIMS APPENDIX

8. A computer program product residing on a computer usable medium for processing structured assembly language, said computer program product comprising:

program code means for implementing a state machine having an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state;

program code means for transitioning from said IF state or said ELSE_IF state to said SETUP_IF state, in response to recognizing a SETUP_IF clause; and

program code means for transitioning from said SETUP_IF state to said ELSE_IF state, in response to recognizing an ELSE_IF clause.

9. The computer program product of Claim 8, wherein said computer program product further includes program code means for transitioning from said IF state to said ELSE state, in response to recognizing an ELSE clause.

10. The computer program product of Claim 8, wherein said computer program product further includes program code means for transitioning from said IF state to said END_IF state, in response to recognizing an END_IF statement.

11. The computer program product of Claim 8, wherein said computer program product further includes program code means for transitioning from said IF state to said ELSE_IF state, in response to recognizing an ELSE_IF clause.

12. The computer program product of Claim 8, wherein said computer program product further includes program code means for transitioning from said ELSE state to said END_IF state, in response to recognizing an END_IF statement.

13. The computer program product of Claim 8, wherein said computer program product further includes program code means for transitioning from said ELSE_IF state to said END_IF state, in response to recognizing an END_IF statement.

14. The computer program product of Claim 8, wherein said computer program product further includes program code means for transitioning from said ELSE_IF state to said ELSE state, in response to recognizing an ELSE clause.

15. A data processing system having an assembler for processing structured assembly language, said data processing system comprising:

a state machine having an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state;

means for transitioning from said IF state or said ELSE_IF state to said SETUP_IF state, in response to recognizing a SETUP_IF clause; and

means for transitioning from said SETUP_IF state to said ELSE_IF state, in response to recognizing an ELSE_IF clause.

16. The data processing system of Claim 15, wherein said data processing system further includes means for transitioning from said IF state to said ELSE state, in response to recognizing an ELSE clause.

17. The data processing system of Claim 15, wherein said data processing system further includes means for transitioning from said IF state to said END_IF state, in response to recognizing an END_IF statement.

18. The data processing system of Claim 15, wherein said data processing system further includes means for transitioning from said IF state to said ELSE_IF state, in response to recognizing an ELSE_IF clause.

19. The data processing system of Claim 15, wherein said data processing system further includes means for transitioning from said ELSE state to said END_IF state, in response to recognizing an END_IF statement.

20. The data processing system of Claim 15, wherein said data processing system further includes means for transitioning from said ELSE_IF state to said END_IF state, in response to recognizing an END_IF statement.

21. The data processing system of Claim 15, wherein said data processing system further includes means for transitioning from said ELSE_IF state to said ELSE state, in response to recognizing an ELSE clause.

22. An assembler residing in a data processing system for processing structured assembly language, said assembler comprising:

means for implementing a state machine having an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state;

means for identifying a SETUP_IF clause;

means for associating said identified SETUP_IF clause with an ELSE_IF clause having a test condition; and

means for inserting instructions from said identified SETUP_IF clause prior to the test condition of said ELSE_IF clause where said ELSE_IF clause logically follows a prior IF clause or a prior ELSE_IF clause.

23. A computer program product residing on a computer usable medium for processing structured assembly language, said computer program product comprising:

program code means for implementing a state machine having an IF state, an ELSE state, an END_IF state, an ELSE_IF state, and a SETUP_IF state;

program code means for identifying a SETUP_IF clause;

program code means for associating said identified SETUP_IF clause with an ELSE_IF clause having a test condition; and

program code means for inserting instructions from said identified SETUP_IF clause prior to the test condition of said ELSE_IF clause where said ELSE_IF clause logically follows a prior IF clause or a prior ELSE_IF clause.